Article

Not peer-reviewed version

Privacy-Preserving Federated Learning-Based Intrusion Detection Technique for Cyber-Physical System

Syeda Aunanya Mahmud , Nazmul Islam , Zahidul Islam , Ziaur Rahman * , Sk. Tanzir Mehedi *

Posted Date: 9 September 2024

doi: 10.20944/preprints202409.0495.v1

Keywords: Federated Learning (FL); Intrusion Detection System (IDS); Internet of Things (IoT); cyber threats; privacy-preserving; FedAvg

Article

Privacy-Preserving Federated Learning-Based Intrusion Detection Technique for Cyber-Physical System

Syeda Aunanya Mahmud [1,‡], Nazmul Islam [1,‡], Zahidul Islam [1], Ziaur Rahman[2,*] and Sk. Tanzir Mehedi [2,3,*]

[1] University of Information Technology and Sciences (UITS), Dhaka, Bangladesh
[2] School of Computer Science, Queensland University of Technology, Brisbane, Australia
[3] Department of Computer Science and Engineering, University of Asia Pacific, Dhaka, Bangladesh
[*] Correspondence: rahman.ziaur@qut.edu.au (Z.R.); s.tanzir@qut.edu.au (S.T.M.)
[†] These authors contributed equally to this work.

Abstract: The Internet of Things (IoT) has revolutionized various industries, but the increased dependence on all kinds of IoT devices and the sensitive nature of the data accumulated by them pose a formidable threat to privacy and security. While traditional Intrusion Detection Systems (IDSs) have been effective in securing critical infrastructures, the centralized nature of these systems raises serious data privacy concerns, as sensitive information is sent to a central server for analysis. This research paper introduces a Federated Learning (FL) approach designed for detecting intrusions in diverse IoT networks that addresses the issue of data privacy by ensuring that sensitive information is kept in the individual IoT devices during model training. Our framework utilizes the Federated Averaging (FedAvg) algorithm, which aggregates model weights from distributed devices to refine the global model iteratively. The proposed model manages to achieve above 90% accuracy across various metrics, including precision, recall, and f1-score, while maintaining low computational demands. The results show that the proposed system successfully identifies various types of cyber attacks, including DoS, DDoS, Data Injection, Ransomware, and several others showcasing its robustness. This research makes a great advancement to the intrusion detection systems by providing an efficient and reliable solution that is more scalable and privacy friendly than any of the existing models.

Keywords: Federated Learning (FL); Intrusion Detection System (IDS); Internet of Things (IoT); cyber threats; privacy-preserving; FedAvg.

MSC: 68T07

1. Introduction

In today's rapidly evolving technological landscape driven by the Industry 4.0 revolution, security has become a crucial viewpoint of any reliable system. As industries increasingly adopt innovative solutions, the integration of smart Internet of Things (IoT) devices has evolved from being merely advantageous to absolutely essential for modern businesses. These IoT devices offer diverse benefits, including enhanced connectivity, seamless communication, and improved decision-making capabilities, which have made them indispensable in contemporary enterprise ecosystems. As the number of IoT devices continues to grow exponentially, which is projected to reach 25.1 billion by 2025 as per the researchers' estimation, also introduces a host of security challenges that cannot be overlooked [1,2]. Unlike traditional computers, IoT devices often use simplified security protocols or weak encryption algorithms to reduce power consumption and cost. Also, these devices typically have limited storage capacity, making it difficult to implement complex security measures that require large amounts of data. These factors combined make it challenging to implement traditional security measures on IoT devices. As a result, they are more vulnerable to various types of cyber-attacks.

While creating more advanced systems, the integration of IoT devices with Cyber-Physical Systems (CPSs) significantly amplifies security risks. In an integrated system, CPSs utilize IoT components to interact with the physical world, monitoring and controlling various processes in industries. IoT devices collect and transmit data about machinery performance, contributing to the overall functionality of the system. However, this increased complexity also exposes CPSs to a wider range of cyberattacks. Attackers can exploit vulnerabilities in the physical or digital components of a CPS to gain unauthorized access to the system. Sensitive data collected or processed by CPSs can be compromised if the system's security is breached. These risks can have significant consequences, including disruptions to operations, financial losses, and potential physical harm. Therefore, it is essential to implement robust security measures to protect CPSs and the IoT devices integrated with them.

Given the growing threats and vulnerabilities in IoT networks, there is a critical need for advanced intrusion detection systems (IDSs) capable of effectively identifying and mitigating various cyberattacks. Although traditional Machine Learning (ML) and Deep Learning (DL) approaches have been used for intrusion detection, Deep Learning (DL) shows greater potential as a solution to secure the ever-growing Internet of Things (IoT) landscape [3]. Most existing intrusion detection systems (IDSs) frequently focus on detecting a limited number of cyberattacks and profiling specific devices, often neglecting automated attack type categorization and overall system reliability [4]. Furthermore, these models typically require extensive training data, and their accuracy can significantly diminish when training data is insufficient.

To address these issues, the growing IoT ecosystem requires more adaptable and reliable solutions. In this case, Federated Learning (FL) models have shown promising results in managing large-scale datasets from diverse IoT devices, offering a potential path forward. Unlike traditional models, FL can train on decentralized data from various sources while minimizing computational overhead, thus maintaining system efficiency. [5]. When evaluating the attack detection capabilities of the IDS, three key metrics are considered: accuracy, efficiency, and adaptability. Accuracy refers to the system's ability to correctly identify and differentiate between normal and malicious activities, minimizing false positives and negatives. Efficiency refers to the IDS's capability to perform intrusion detection with minimal computational overhead, ensuring quick response times and low resource consumption. Adaptability is the system's ability to handle an increasing number of data sources from a wide range of IoT sensors, allowing it to maintain high detection rates even as the network scales and evolves. Considering all the challenges, we propose five deep learning (DL) architectures, each utilizing the Flower-based federated learning (FL) framework, to enable efficient intrusion detection in IoT environments. Figure 1 illustrates our proposed approach, a federated learning-based intrusion detection technique for cyber-physical systems (CPSs). Here, the final model is selected based on optimal prediction performance on the set of heterogeneous IoT sensor data. We trained all five deep learning models on an unbiased training dataset and validated them using a separate validation dataset. We then comprehensively compared the models' detection capabilities across various discrete parameter types.

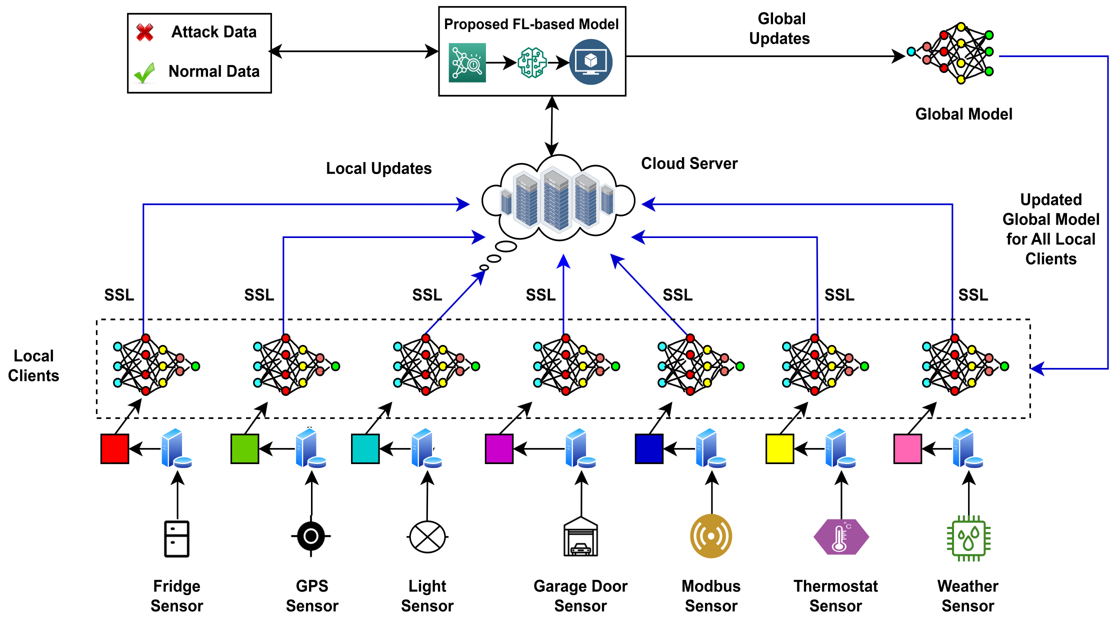

Figure 1. Overview of the proposed Federated Learning framework for intrusion detection

Our proposed approach utilizes seven IoT devices equipped with local clients trained on diverse sensor data. During training, local clients generate weights, which are then sent to the cloud server via a secure communication channel using Secure Socket Layer (SSL). Those weights are then aggregated by the cloud server using the FedAvg algorithm. The resulting global updates are then distributed to local clients, enhancing their detection capabilities. This iterative process continues until the global model ensures optimal accuracy in classifying normal and attack data. Compared to the existing approaches, our model achieves competitive results across several key performance metrics. The overall accuracy of the proposed detection model exceeds 90% while maintaining low computational demands. In addition, the model significantly improves other performance metrics. Our approach outperforms all baseline approaches, achieving top-performing metrics across the five deep learning algorithms with a precision score of 0.87%, a recall score of 0.87% and an f1 score of 0.87%, and also reducing training loss of 20%. These evaluations have proven that the proposed model is robust, more efficient, and performs better. This paper makes significant contributions to the field of intrusion detection for heterogeneous IoT networks. The main contributions of this paper are outlined as follows:

- In this work, a deep learning architecture based on federated learning has been proposed for efficient intrusion detection. This model successfully identifies a variety of attacks against heterogeneous IoT networks, including denial of service (DoS), distributed denial of service (DDoS), data injection, man-in-the-middle (MITM), backdoor, password cracking attacks (PCA), scanning, cross-site scripting (XSS) and ransomware attacks.
- The tests have been carried out by using real-world data from seven different IoT sensors, providing a diverse and realistic training dataset. By analyzing real-world sensor data streams, we identified optimal features for our supervised learning framework, which speed up training and improve performance metrics.
- Finally, a broad variety of deep learning architectures have been employed in performance evaluation, demonstrating that the proposed model is more sustainable and effective than similar models developed previously.

This research paper is further structured as follows. First, Section 2 presents an overview of Privacy Preserving in FL, addressing key privacy challenges in FL, and exploring possible solutions. This includes understanding Federated Learning in terms of Privacy Preservation and categories of Federated Learning. Section 3 provides the background and related work in this field. In Section

4, we describe the methodology for our proposed privacy-preserving FL-based IDS, including the problem statement, SSL mechanism, the workflow of the proposed model, aggregation functions, and the proposed architecture. The model workflow is further broken down, exploring each stage: initialization, local training on sensor data, parameter encryption, parameter aggregation, local model update, and overall computational complexity. Model implementation and model evaluation are discussed in Section 5. Furthermore, an analysis of the results is provided in Section 6. Section 7 presents a summary and possible future research directions.

2. Privacy Preserving FL

Federated Learning is an advanced machine learning model recommended by the Google developer team that prioritizes privacy preservation and the security of user data. It protects user data by enabling collaborative model training across decentralized devices, such as sensors and smartphones, without centralizing raw data. This method addresses the challenges associated with a central server or device to train a shared global model, thus enhancing the privacy and security of user data [6–8].

Initially, they implemented the federated learning model (FL) to train the machine learning model (ML) based on data distributed across various mobile devices. In this approach, each client (e.g., sensors, smartphones) independently trains models on their local datasets, ensuring the security of user data during the training process[9,10]. With the continuous increase in sensor devices, a large amount of data has been generated in recent years. As a result, it is recommended to store data locally and perform computations directly on these devices using edge computing, which ensures privacy and security issues as well as the problems of increasing the extensible computing capabilities of the cloud server [9,11,12]. Advances in local device storage and processing capabilities have further enabled the efficient implementation of this mechanism [8]. To maximize this potential, FL acts as a promising approach that can enable direct analysis of data on local or remote devices [13]. Traditional machine learning typically involves centralizing data from various sources for model training. In contrast, the FL mechanism can be widely applicable to large-scale applications in artificial intelligence, medical technology, and IoT, primarily to enhance data privacy and security. [14,15].

Studies have shown that most large-scale service providers have already incorporated FL technology [9,10,12,14,15]. Various types of devices are increasingly equipped with numerous sensors, enabling real-time acquisition, aggregation, response, and adaptation to new sensitive sensor data in a real-time environment [16,17]. To accurately predict the intrusion, sensor devices require the latest models trained on a large dataset of attack and non-attack data to operate safely and make real-time predictions. However, building aggregate models in such scenarios is challenging due to privacy and security concerns surrounding sensitive data and limited device connectivity [18].

Due to the heightened sensitivity of sensor data, it is necessary to ensure the privacy and security of sensor data when analyzing this information. Federated learning (FL) technology facilitates model training in this context, allowing for the rapid adaptation of models while ensuring user privacy and security [15]. In this approach, data are stored securely on local devices [19,20]. This makes FL an ideal approach for instruction detection, which often involves managing large volumes of sensitive data [18,20]. This FL technique also enables secure and private collaborative learning among organizations where sensor data privacy is a primary concern.

3. Related Work

While exploring the latest related literature, we found several projects that had similar motivations but used different strategies. The following discussion highlights these works before presenting our proposed approach.

Federated Learning (FL) is emerging as the leading solution for intrusion detection in Industrial Internet of Things (IIoT) networks, surpassing the capabilities of centralized models. This approach not only enhances data privacy and security, but also provides overall performance improvements. The IDS proposed for the Industrial Internet (IoT) by Pedro Ruzafa-Alcázar et al. also uses this dataset to address privacy concerns when updated gradients/weights are exchanged during the training period; the analysis uses differential privacy techniques (DP). The model is built upon Multinomial Logistic Regression, and evaluated using the FedAvg and Fed+ aggregation methods [21].

Chatterjee and Hanawal [22] introduced a pioneering hybrid ensemble model (PHEC) for Intrusion Detection Systems (IDS) in the context of IoT security. Their model demonstrated good performance in centralized environments, and its extension to a federated learning framework retained comparable effectiveness, confirming its promise to address challenges in decentralized and privacy-sensitive IoT environments. Additionally, they introduced a noise-tolerant version of PHEC to handle label noise. Their model demonstrated exceptional performance, effectively distinguishing genuine threats while minimizing false alarms, even in noisy conditions.

Lazzarini et al. [23] conducted a sophisticated study using Federated Learning (FL) for Intrusion Detection Systems (IDS) in IoT environments. Their work supports the effectiveness of FL in preserving data privacy while achieving competitive accuracy, precision, recall, and F1 scores compared to centralized approaches using shallow artificial neural network (ANN) and federated averaging (FedAvg).

Zhang et al. [24] introduced FeDIoT, a platform that uses Federated Learning (FL) directly on devices to detect anomalies in IoT traffic. Using the N-BaIoT and LANDER datasets, their study demonstrated that Federated Learning (FL) could be used to effectively identify and detect network attacks. This technique involved clients within the network running a specific FL model that used a method called Auto-Encoder.

An outstanding contribution by Chen et al. [25] proposed a Federated Learning-based Attention Gated Recurrent Unit (FedAGRU) for intrusion detection within wireless edge networks. This approach outperformed the FedAvg aggregation model in terms of communication efficiency during model updates, while achieving higher accuracy in detecting attacks compared to a centralized Convolutional Neural Network (CNN) model.

Aashmi and Jaya [26] introduced an intrusion detection system utilizing Federated Learning (FL) within Jungle Computing (JC), integrating various computing platforms. This system effectively detects intrusion attacks without human intervention, achieving 96% accuracy with low False Positive Rates. The authors emphasized Jungle Computing's adaptability to diverse computing paradigms, which enhancing the system's efficiency and robustness.

Truong Thu Huong et al. [27] presented FedeX, a Federated Learning-based architecture for Explainable Anomaly Detection in Industrial Control Systems (ICS) within Smart Manufacturing. Outperforming 14 existing solutions, FedeX excels in detection metrics, run-time efficiency (7.5 minutes), and lightweight deployment on resource-constrained edge devices (14% memory consumption). The architecture integrates the Variational Autoencoder (VAE) and Support Vector Data Description (SVDD) with Federated Learning. It ensures distributed anomaly detection in smart factories while addressing interpretability through explainable Artificial Intelligence (XAI).

Popoola et al. [28] introduced Federated Deep Learning (FDL), a method for detecting zero-day botnet attacks on IoT edge devices, to address data privacy concerns. FDL, employing a deep neural network (DNN) architecture with a Federated Averaging (FedAvg) algorithm, proved superior over centralized deep learning (CDL), localized deep learning (LDL) and distributed deep learning (DDL) methods. The results demonstrated high classification performance, guaranteed data privacy, low communication overhead, minimal memory space requirements, and low network latency.

Kun Li et al. [29] introduced a distributed Network Intrusion Detection System (NIDS) using Federated Learning (FL) to address security and privacy challenges in satellite-terrestrial integrated networks (STINs). They designed a typical STIN topology and developed security datasets for satellite and terrestrial networks. To increase resistance to attack and resource allocation, the authors introduced a satellite network topology optimization algorithm and an FL adaptation algorithm for STIN. Their distributed NIDS, outperformed traditional methods in detecting malicious traffic, reducing packet loss, and optimizing Central Processing Unit (CPU) utilization.

Fangyu Li et al. [30] proposed a federated sequence learning (FSL) approach to address the challenges of modeling varying time series data in IIoT while prioritizing data security. FSL, under a federated framework, constructs a collaborative global model without compromising local data integrity, capturing intrinsic industrial time-series responses, and considering data heterogeneity among distributed clients. Experiments on classic distributed datasets and real IIoT attack detection scenarios demonstrate FSL's capability to accurately model data heterogeneity and improve performance in attack detection for industrial time-series sensory data compared to existing methods, making it promising for real IIoT applications.

Mohanad Sarhan et al. [31] proposed a collaborative cyber threat intelligence sharing scheme that uses Federated Learning to allow multiple organizations to train and evaluate a Machine Learning-based intrusion detection system. The framework was tested with two key datasets, NF-UNSW-NB15-v2 and NF-BoT-IoT-v2, and evaluated under centralized and localized training scenarios. The results show the framework's effectiveness in creating a robust ML model that accurately classifies various network traffic types from multiple organizations, all while preserving data privacy.

Roberto Doriguzzi-Corin et al. [32] introduced FLAD (Adaptive Federated Learning Approach to DDoS Attack Detection), a novel federated learning (FL) solution for cybersecurity that addresses the challenge of updating deep learning models with new attack profiles without sharing test data. Unlike traditional FL, which requires test data to monitor model performance, FLAD dynamically allocates more computational resources to members with harder-to-learn attack profiles. The authors demonstrated that FLAD outperforms existing FL algorithms in terms of convergence time and accuracy across diverse, unbalanced datasets. Additionally, FLAD's robustness is validated in a realistic scenario where the model is repeatedly re-trained to incorporate new attack profiles, enhancing detection capabilities for all federation members. Table 1 shows a summary of recent research on FL-based IDSs for IoT applications.

Considering the specifics of different IoT applications, this paper significantly enhances the existing federated learning model. The primary aim is to strengthen the model's performance in effectively detecting a diverse array of complex cyber-attacks.

Table 1. Summary of recent research on FL-based IDSs for IoT applications.

Ref.	Detection Category	Accuracy	Dataset	ML/DL Models	FL Implementation	Aggregation Function	Validation Strategy
[31]	Intrusion Detection System	>85%	NF-UNSW-NB15-v2	LSTM, DNN	TensorFlow	FedAvg	Empirical (2 Devices) 9 Categories
[30]	Cyberattack Detection System	>93%	Edge-IIoTset	CNN, LSTM, FSL	-	FedProx	Empirical (4 Devices) 7 Categories
[32]	DDoS Attack Detection	>96%	CIC-DDoS2019	MLP	TensorFlow	FedAvg, FLAD, FLDDOS	Empirical (13 Devices) 13 Categories
[29]	Intrusion Detection System	>83%	SAT20, TER20	CNN	OpenStack, TensorFlow	FedSGD	Simulation
[22]	Intrusion Detection System	>88%	NSL-KDD	KNN, RF, MLP	-	FedStacking	Empirical (4 Devices) 4 Categories
[21]	Intrusion Detection System	>88%	TON_IoT	Multinomial Logistic Regression	-	FedAvg, Fed+	Empirical (7 Devices) 9 Categories
Proposed	Intrusion Detection System	>90%	TON_IoT	DNN, LSTM, GRU, FCN, LeNet	Flower, TensorFlow	FedAvg	Empirical (7 Devices) 9 Categories

4. Proposed Privacy-Preserving FL-Enabled IDS for IoT/IIoT

4.1. Problem Statement

The traditional intrusion detection system has several limitations, such as limited concerns about data privacy and security, challenges in scalability and efficiency, issues with data heterogeneity in network environments, and difficulties with resource constraints in IoT networks, particularly in providing efficient real-time intrusion detection.

This research is motivated by those key factors driving the need for an advanced intrusion detection system that overcomes the constraints of traditional techniques. Most centralized IDSs compromise the privacy and security of client data because they require sensitive data to be aggregated into a centralized repository. To address this, we develop a FL-based intrusion detection system, where our system explores a decentralized approach to mitigate these limitations by keeping data localized.

Traditional IDSs often fail to identify new and emerging cyberattacks because they are designed to identify only a limited set of known cyberattack patterns. In this situation, the goal of our proposed model is to efficiently detect and classify a broad spectrum of cyber-attacks, including DoS, DDoS, data injection, MITM, backdoor, PCA, scanning, and ransomware. These attacks are particularly difficult to detect while maintaining low computational complexity. Traditional IDSs face challenges of scalability and efficiency due to the increase in data volume and network traffic [5]. Our FL-based model addresses these issues by distributing the learning process, enhancing scalability and efficiency. In a network, different types of sensor devices generate different types of data with a variety of features and distributions. Our model can handle heterogeneous data and provide efficient intrusion detection in diverse networks.

Networks with limited resources, such as IoT networks, need vast data to build complex and efficient IDSs. However, with limited resources, IoT networks cannot run complex IDS [33]. Our proposed system leverages federated learning as a resource-efficient and lightweight alternative to overcome these limitations. To build a network with effective security, a real-time intrusion detection system is needed. Our proposed model provides this capability without compromising accuracy. Existing research has yet to do much profiling of devices [5]. Our approach takes into account the behavior of seven IoT devices, such as a fridge, GPS tracker, motion light, garage door, modbus, thermostat, and weather sensor. The integrity of the proposed federated learning (FL) based model is enhanced for these considerations.

Most suggested IDSs did not consider reliability [3]. Our FL-based model provides reliability and does not compromise the accuracy of classifying different types of attacks. IoT applications typically operate on lightweight communication protocols, as well as on limited computing and storage capacity [[34]. However, traditional security systems require complex computing power with massive storage; as a result, this system can not be used directly for IoT applications [33]. In addition, new threats and vulnerabilities to the IoT increase rapidly due to traditional intrusion detection systems not being able to handle this new attack. IoT applications run inside the network, and IoT applications are different from each other by their nature of work in the environment [34,35]. Traditional intrusion detection systems can only identify attacks outside the network; as a result, these old systems are not strong enough to fight targeted attacks and related anomalies. In contrast, our proposed FL-based intrusion detection system efficiently addresses these challenges.

Many earlier IDS proposals did not focus on strengthening the performance for diverse attack detection. To address these issues, we propose a privacy-preserving Federated Learning-based intrusion detection technique for Cyber-Physical Systems (CPS). In this approach, local models are trained on each IoT device, which can detect whether the data are normal or an attack. Then, the local models generate weights based on their data, which are sent to the server via a secure connection where the global model resides. The global model aggregates these weights to produce a new set of weights, which is sent back to the local models for the next round of training. This process continues iteratively until the global model converges to a set of weights that can accurately detect both normal and attack data.

4.2. Federated Averaging (FedAVG)

In federated learning, federated averaging is a primary aggregation method that allows decentralized devices to collaboratively train a model without transferring raw data [36]. This method plays a vital role in combining the knowledge gathered from various sources and constructing an improved global model.

Each participating device in federated learning uses its own local data to train the local model, without sharing the raw data with the central server. The local dataset of each device can be different in terms of how it is distributed, size, and data quality, allowing the local model to learn from different perspectives [4]. To be more specific, each participating device i trains a local intrusion detection model using its locally available data, where the local model parameters θ_i are optimized to minimize a local loss function $L_i(\theta_i)$, which is specific to the data available on that device. Local training can be expressed as

$$\Delta w_i = \arg\min_{\theta_i} \mathcal{L}_i(\theta_i)$$

After local model training, each device computes a model update $\delta\theta_i$ based on the changes in its local model parameters. These model updates are sent to the central server, which is crucial for preserving data privacy, as raw data remains on the local devices and is not directly shared [4]. The central server collects these model updates from all participating devices to create global updates for the local model. The server then aggregates these local model updates and creates the global updates using the FedAvg algorithm. FedAvg is a weighted averaging technique that gives each model update an appropriate weight based on factors like how much data the device has and how well the local model performs [37]. The aggregated update $\delta\theta$ is computed as,

$$\Delta\theta_{avg} = \frac{1}{N} \times \sum_{i=1}^{N} \Delta\theta_i$$

Here, N is the total number of participating devices. In this aggregation technique, devices with more data or better local models may have more impact on global updates or the model [37]. In each round, after the central server aggregates the model updates, it applies the aggregated update to the global model parameters θ_g to obtain the updated global model parameters θ,

$$\theta = \theta_{avg} + \theta_g$$

The updated global model θ is sent back to the participating devices. These devices use the updated global model as a starting point for their local model training in the next round. The process of local training, model update sharing, and aggregation using FedAvg is repeated for a predefined number of rounds to refine the global model [4].

4.3. SSL Mechanism

Secure Sockets Layer (SSL) ensures a secure communication channel between the central server and decentralized devices during federated learning model updates. SSL encrypts data using strong encryption methods, making it unreadable to anyone without the decryption key[38]. This encryption protects sensitive data from unauthorized access. SSL also authenticates the central server and decentralized clients to verify that only authorized clients can engage in federated learning. SSL uses a system of digital certificates and public and private keys to ensure that the central server and the decentralized devices are who they say they are. The digital certificate is issued by an authentic organization called Certificate Authorities (CAs). During the SSL handshake process, this digital certificate allows local clients and cloud server to participate in federated learning. This method of authentication prevents unauthorized clients from accessing the system and manipulation with global updates to the model [39]. Furthermore, SSL verifies the integrity of model updates during transmission to ensure that they have not been altered. SSL uses cryptographic hashes and algorithms to ensure the integrity of the data. To ensure that data is not altered while in transmission, each segment of data is converted into a unique code called a hash. The hash is then sent along with the data itself. When the data arrive at their destination, they are hashed again, and the new hash is compared to the original hash. If the hashes match, it means that the data has not been changed [40]. This is necessary to ensure the accuracy of the federated learning model.

4.4. *Workflow of the Proposed Model*

The proposed model aims to create a reliable and privacy-preserving FL model. The model's complete workflow includes model initialization, local model training by sensor clients, model parameter encryption by sensor clients, model parameter aggregation by cloud server, local model update by sensor clients, and overall computational complexity of the model.

4.4.1. Model Initialization

At first, a trusted key management authority generates a pair of cryptographic keys, which are the public key and the private key. These keys are generated by using the key generation method with a security parameter. The trusted key management authority then establishes a secure and protected communication channel between the cloud server and each sensor client. The cloud server also selects an array and sets its initial parameters for the federated learning algorithm. Next, each sensor client reports the size of its personal data resource to the cloud server, and then the cloud server calculates each sensor client's contribution ratio. Finally, the cloud server sets the index of the first communication round at one, which specifies the beginning of the iterative federated learning process. In subsequent rounds, the model parameters are updated using data from decentralized clients.

4.4.2. Local Model Training by Sensor Clients

Every sensor client trains a deep learning framework locally, utilizing their own data resource while keeping everything personal. It starts with the cloud server distributing the parameters of the initial model, which consist of learning rate, moments estimators, numerical factorization for stabilization, loss function, batch size, previous model parameters, a constant, and data for training. The local model is trained until the loss function converges. In this stage, the first and second-moment variables are initialized to zero. Then, The dataset is divided into batches of equal size, and the local model parameters are set based on the previous round's parameters. Furthermore, for each batch of data resources compute the gradient biased first and second moment estimates respectively. Later on, the one moment bias-corrected estimate and the second moment are obtained along with the bias-corrected learning rate. Finally, the local model parameters are further modified, and the new model parameters are sent to the cloud server. To avoid computational complexity and impacts on real-time demands, this training process is performed offline.

4.4.3. Encryption of Model Parameters by Sensor Clients

This process ensures the privacy of the parameters of the local deep learning (DL) model, encrypting these parameters before sending them to the cloud server. Once a sensor client has trained its local DL model and generated the model parameters, it encrypts these parameters using the public key with the encryption method. Then, the encrypted parameters of the local DL model are uploaded to the cloud server by each sensor client.

4.4.4. Aggregation of Model Parameters by Cloud Server

The cloud server computes each client contribution ratio and aggregates these ratios with the encrypted parameters. Finally, the aggregated parameters are sent back to all the clients through a secure channel by the cloud server so that all the clients are in a position to modify their local parameters by the new global model parameters.

4.4.5. Local Model Updating by Sensor Clients

This process shows how the sensor clients perform local model updates with the aggregated parameters provided by the cloud server. In the first step, each client decrypts the encrypted parameters with the help of a private key corresponding to the public key used for encryption. Once clients decrypt the encrypted parameters, they modify their local model to take into account global changes. The index value is increased one by one after the completion of each successful update operation. The process repeats between the cloud server and the clients for a selected number of rounds (an empirically determined threshold), resulting in an efficient DL-based model.

4.4.6. Overall Computational Complexity of the Model

At every communication round that has been completed, all the clients are required to perform both parameter encryption and parameter decryption operations. These two tasks involve a certain number of exponentiations and several multiplication operations. The level of computation for each client corresponds to the parameter in the local deep learning model. Moreover, the server needs to perform various multiplication operations on every communication round that is successfully completed when it is aggregating the model parameters as well as the contribution ratios from the clients who took part.

4.5. Proposed Architecture

The block diagram in Figure 2 shows that the proposed model has two main parts: the local clients and the cloud server. The local clients are further divided into two subcomponents: the model training part and the model validation part.

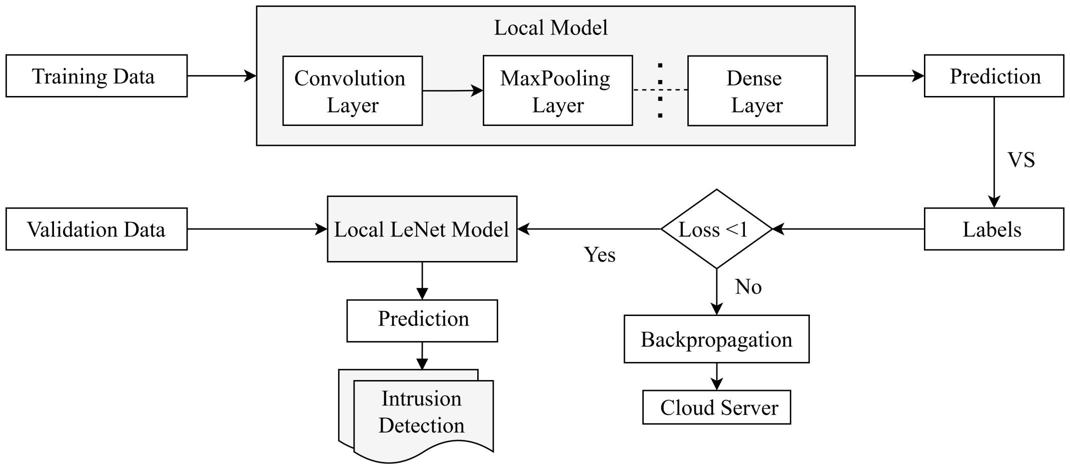

Figure 2. Block diagram of the proposed model with a local client and a cloud server

After preprocessing the raw sensor data, it is applied to the local client model for training. Subsequent empirical investigations identify the most important parameters of this model. Randomly selected training and validation datasets are utilized to train and validate the local client model. At the training stage, the local model parameters have been back-propagated to the cloud server if the local model loss is not negligible. After collecting each of the local model parameters, the cloud server aggregates the weights, averages the parameters, and then sends them back to all local clients' models so that they can update their parameters' weight to minimize the loss. The process is repeated until the loss is as small as possible; at this point, the best performance indicators are used to choose the final model. Table 2 shows the hyperparameters of the proposed model.

Table 2. The hyper-parameters of the local model.

Hyper-parameters	Value/Function
Number of Hidden Layers	3
Units in Hidden Layers	32, 64, 256
Batch Size	512
Epochs	5
Hidden Layer Activation Function	ReLu
Output Layer Activation Function	Softmax
Dropout	N/A
Optimizer	Adam
Learning Rate	0.01
Decay	0.005
Loss Function	Categorical Cross-entropy

Local client devices consist of multiple layers with hyperparameters, including the input layer. In this layer, we give sensor data as input to our model. This layer contains a number of neurons that correspond to the total number of features. The input data are preprocessed, which is identical for all clients. This input is then fed into subsequent layers for further processing. Following the input layer, a convolution layer transforms the encoded sensor data into a vector of features [2]. This trainable layer applies a sliding window across the encoded sensor data [41]. The model uses three convolution layers in our model. The first layer combines 32 filters with a kernel size of 5, followed by the second layer with 64 filters, and the third with 256 filters, all having the same kernel size of 5. Each of these layers is a one-dimensional convolutional layer, followed by BatchNormalization() and a Rectified Linear Unit (ReLU) activation function, collectively referred to as the feature smoothing layer (FSL). Following the convolution layer, a non-trainable pooling layer is incorporated to modify the feature map's dimensions. Various pooling techniques exist, including maximum pooling and average pooling. We implemented max-pooling layers, identifying the neuron with the most significant activation value. This process significantly reduces the number of parameters in a deep model while facilitating a class activation map generation [35], which allows an interpretation of the learned features.

A dropout layer is added after the second max pooling layer to enhance the model's robustness and prevent overfitting. This layer randomly deactivates a specified percentage of neurons or connections during training, as defined by a dropout ratio of 0.10 [1]. The data must undergo a flattening process to facilitate its incorporation into the subsequent layer, transforming the multidimensional output into a one-dimensional array. This layer reshapes the preceding layers' output into a single, elongated feature vector, which is then connected to the next layer. The next layer is a fully connected layer that combines information from all the previous layers. This layer helps to learn complex relationships between features, making final prediction easier. A dense layer is used to classify attacks based on the information extracted from the preceding layers. This layer establishes direct connections with the preceding layer, effectively combining and extracting higher-order features from the input sequence. The ReLU activation function, which introduces non-linearity into the model, is utilized within this layer. Finally, the softmax layer is connected to a dense layer. This layer serves the crucial function of determining the probability of each classification or label, effectively predicting whether a new attack is present or not. In this layer, the number of neurons corresponds to the total classes present in the dataset [42,43].

5. Model Implementation and Evaluation

5.1. Environment Setup

In this study, we conducted experiments within a controlled computing environment to ensure the reliability and consistency of our results. We employ a total of eight computers, consisting of one central server and seven client computers, all with similar hardware specifications. Our central server was equipped with an 11th Gen Intel(R) Core(TM) i7-1185GRE (2.80GHz) Central Processing Unit (CPU) and a high-performance Intel(R) Iris(R) Xe (1.35 GHz) Graphics Processing Unit (GPU). This setup provided us with the capability to perform a series of analytical tasks on the original dataset. Additionally, we thoroughly explained the training process and offered in-depth information on the chosen models used for our experiments. This approach guaranteed that our research was conducted under standardized conditions.

5.2. Implementation Procedure

Figure 3 illustrates the implementation procedure of a system involving a Key Management Authority (KMA), a switch, a computing server, and multiple clients. The KMA is responsible for managing cryptographic keys, converting plaintext to ciphertext using public keys, and vice versa using private keys. It communicates with the switch, which serves as the central hub that connects all components. The switch directs traffic between the KMA, the computing server, and the clients. The computing server processes data and computations and communicates with the switch. Clients, labeled 1 through N, represent devices connected to the switch. They perform local tasks and interact with the system using their unique IP addresses. This setup ensures secure communication and data processing within the network, using the KMA for cryptographic operations and the switch for efficient data routing.

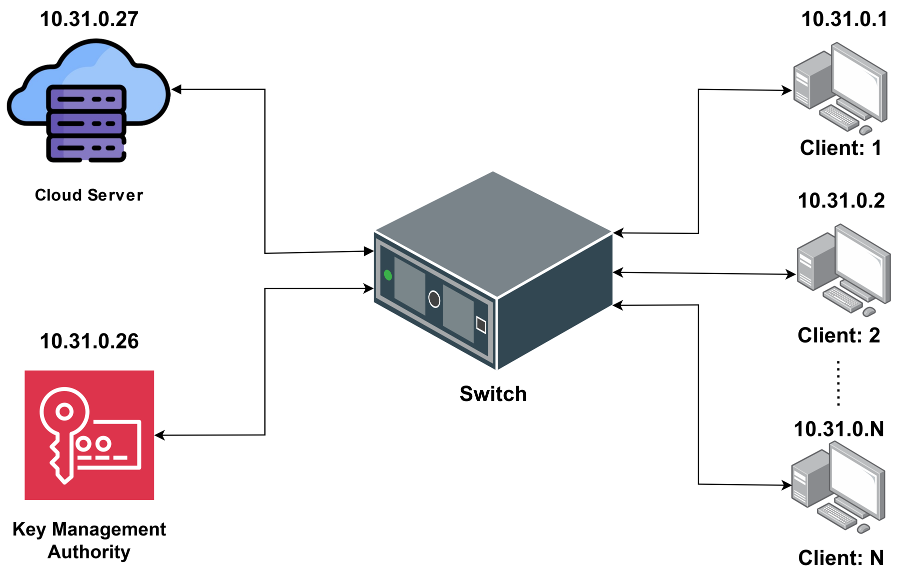

Figure 3. Implementation procedure

5.3. Data Description

The ToN-IoT datasets contain data collected from seven IoT and IIoT sensors, covering various devices such as Fridge, GPS Tracker, Motion Light, Garage Door, Modbus, Thermostats, and Weather sensor. These datasets are labeled to distinguish between normal and attack data. They cover nine types of cyber-attacks, including Scanning, DoS, DDoS, ransomware, backdoor, data injection, Cross-site Scripting (XSS), password cracking, and Man-in-The-Middle (MITM) attacks, systematically launched across various IoT and IIoT sensors within the network [2].

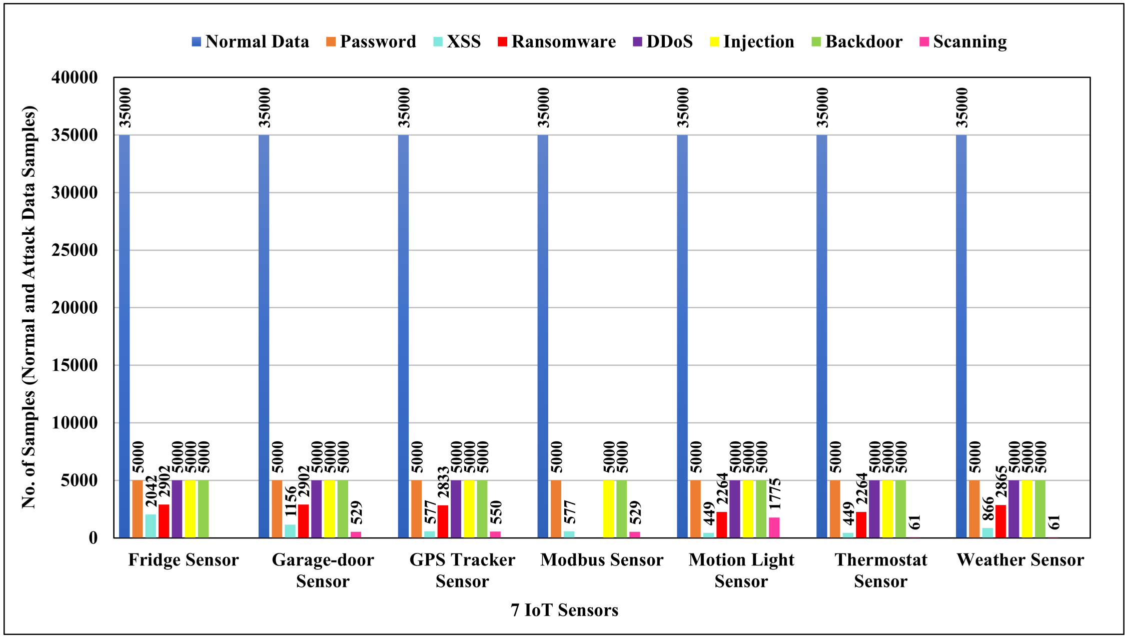

Figure 4. Class distribution of normal and attack data for 7 IoT sensors

The Smart Fridge sensor monitors and adjusts the temperature to maintain optimal conditions. It measures the current fridge temperature and assesses its condition, determining whether it falls within the predefined threshold values for high or low temperatures. The Garage Door sensor enables remote operation for door opening and closing based on two critical states. The first state, known as the "door state," indicates whether the garage door is opened or closed. The second state, "sphone_signal," denotes the status of receiving door signals on a mobile phone, where the signal is either true or false. The GPS Tracker sensor tracks location coordinates, including latitude and longitude. The Smart Motion sensor has two characteristics; first, it monitors motion status; then, it controls light status based on a pseudo-random signal. The Modbus sensor facilitates communication between these devices through a master-slave protocol to transmit various register types, including input, discrete, and holding registers over serial lines. The Smart Thermostat regulates temperature in physical systems through two primary features: current temperature reading and thermostat status, which indicates that the sensor is on or off. Finally, the Weather sensor generates air pressure, humidity, and temperature data [2,21].

5.4. Data Preparation

The data needs to be preprocessed correctly to get better results from the machine learning model in projects. Data preprocessing involves eliminating unnecessary features, dropping missing values, handling imbalance classes, removing white spaces, encoding categorical data, detecting outliers and noise, checking correlation, and normalizing the different ranges of values into a specific range to get better performance from our desired FL model. Two main preprocessing tasks are employed for this model. Data cleaning and preprocessing is the first one, and the second one is feature engineering.

5.4.1. Data Cleaning and Preprocessing

The dataset originates from various sources with different characteristics. It is very susceptible to missing and noisy data due to its size [1].

- **Handling Null Values**: To handle null values in the dataset, we apply the predictive mean matching method to input missing values. This technique involves predicting missing values based on their relationships with other variables and then matching them to observed values to ensure a more accurate imputation.
- **Handling Noise and Outliers Values**: To address noise in the dataset, Rosner's Test is applied.

- **Removing White Spaces from Target Class**: The 'strip()' string method is used to remove leading and trailing whitespace (including spaces) from a string, ensuring that the contents do not start or end with spaces. For example, the 'temperature condition' feature in the fridge sensor dataset, which has white spaces with values like 'high ' and 'low ', has been converted into 'high' and 'low'.
- **Checking Balance Data**: The class Distribution Summary method is used to check balance data, and the Bar Chart ensures that the dataset is balanced. Each individual sensor dataset has been confirmed to be balanced.

5.4.2. Feature Engineering

Feature engineering is a pre-processing procedure in machine learning that turns unprocessed data into features that may be used to build prediction models using either machine learning or statistical modeling. This section discusses necessary steps for feature engineering.

- **Eliminating Unnecessary Features**: It involves selecting and determining which features are relevant to the FL model, thus improving the quality of the feature set. To provide one instance, the 'time_stamp', 'time', 'date', and 'type' features in the fridge sensor dataset are dropped to enhance the model's performance.
- **Encoding Categorical Data**: Categorical values were converted into Numerical format. To provide one instance, the 'light_status' feature in the Motion light sensor dataset, which has qualitative states of 'on' and 'off', has been transformed into their respective numeric values '1' and '0'. This transformation was achieved using the *numconv* library, which applies numerical convolution for label encoding [4].
- **Replacing Boolean Data with Binary Values**: Boolean values were converted into binary format. To provide one instance, the 'sphone_signal' feature in the garage door sensor dataset, which has a 'True' and a 'False', has been converted into '1' and '0'.
- **Normalizing Different Range of Values**: Normalization ensures that numerical features have consistent scales, which can improve the performance of our FL model. Some models may prefer large feature values; however, some features have greater values than others, which causes erroneous performance. These techniques have been used to scale the chosen feature values within the interval [0.0, 1.0] without affecting the properties of the data [44]. The chosen feature values have been scaled within the range using a method known as min-max data normalization, as indicated by the following equation:

$$X_{\text{normalized\_value}} = \frac{(X - X_{\text{min}})}{(X_{\text{max}} - X_{\text{min}})}$$

  where X is the feature's initial value and X_{max} and X_{min} are its maximum and lowest values, respectively.
- **Correlation Checking**: A measurable strategy for calculating the link and gauging the strength of a linear relationship between two factors is correlation analysis. The degree to which a change in one variable affects a change in another is determined through correlation analysis [1].

The strength with which one feature—door state—implies another—sphone_signal—has been measured in this investigation. Figure 5 illustrates how several traits are correlated. In particular, to assess the correlation between all the features, Pearson's product-moment coefficient equation has been used [1].

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

[1] https://www.javatpoint.com/feature-engineering-for-machine-learning

Here, n represents the number of data points in the dataset for both x and y, quantifying how well changes in x and y can be explained by a linear relationship.

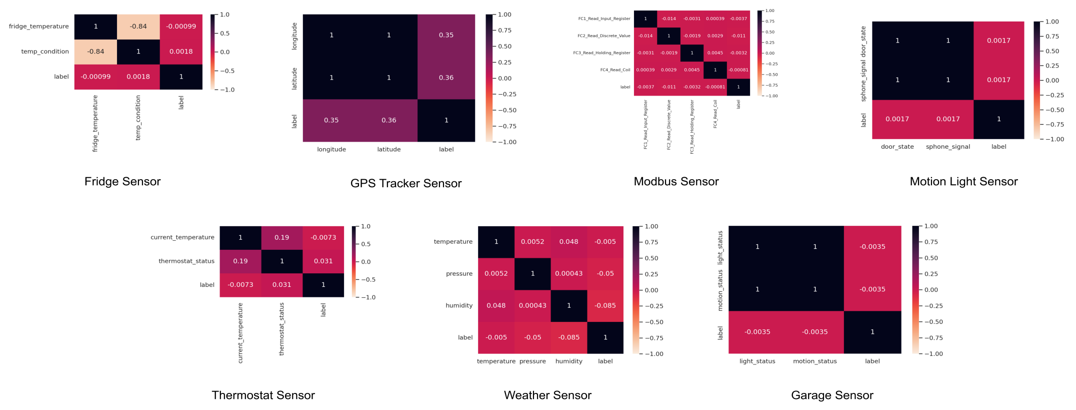

Figure 5. Correlation Visualization.

5.5. Training Process

As discussed in the previous section, we considered seven IoT sensors as seven individual clients and utilized their datasets to train the global model using a federated learning approach. Initially, we have preprocessed the heterogeneous sensor data to enhance the quality and relevance of training. Following preprocessing, the data set was divided into training sets (80%) and testing sets (20%) using the split train test method of the scikit-learn library [1]. This proportion between training and testing data sets has been recognized as the best ratio to avoid the overfitting problem [45].

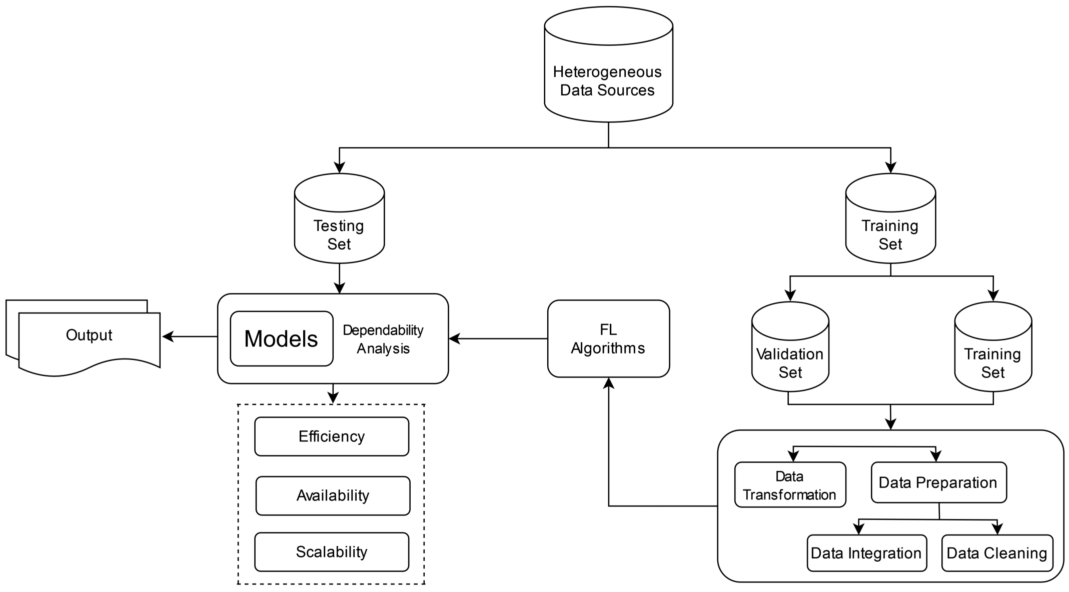

Figure 6. Training Process

5.5.1. Model Selection and Initialization

We have selected 5 Deep Learning (DL) algorithms such as Deep Neural Network (DNN), Long Short-Term Memory (LSTM), Fully Convolutional Network (FCN), LeCun Network (LeNet) and Gated Recurrent Unit (GRU). We chose these DL algorithms because some of their variations have been used successfully to solve classification problems associated with intrusion detection [42]. A federated learning setup with seven clients was established. Each client had a local sensor dataset, and the objective was to collaboratively train a global model collaboratively without sharing raw data. The training process began by initializing the global models for each deep learning architecture (DNN, LSTM, FCN, LeNet, & GRU) with random weights. Then these initial models were distributed to the seven clients.

5.5.2. Training Rounds

Each client independently trained its assigned model (e.g., DNN) using its local sensor dataset. The locally trained models were then aggregated using Federated Averaging (FedAVG) to construct a global model (e.g., global DNN model) that aggregated the collective knowledge derived from the local models of all seven clients. At the end of each training round, the updated global model was shared with all clients who replaced their current local models with it. Each client fine-tuned its local model using the updated global model. This ensured that each client benefited from the collective insights of the entire federated learning setup. The global model performance was further refined by repeating a similar process for 50 rounds to enhance the convergence. Finally, we evaluated all models (DNNs, LSTMs, FCNs, LeNets, and GRUs) on the test data to assess their performance. The hyperparameters of the selected FL models are shown in Table 3.

Table 3. The hyper-parameters of the selected deep learning models.

Parameters	**LeNet**	**DNN**	**LSTM**	**GRU**	**FCN**
Number of Hidden Layers	3	3	4	2	3
Units in Hidden Layers	32, 64, 256	128, 64, 64	128, 256, 512, 256	32, 64	128, 256, 128
Batch Size	512	512	512	512	512
Epochs	5	5	5	5	5
Hidden Layer Activation Function	ReLu	ReLu	PReLu	ReLu	ReLu
Output Layer Activation Function	Softmax	Softmax	Softmax	Softmax	Softmax
Dropout	N/A	3	3	1	N/A
Optimizer	Adam	Adam	Adam	Adam	Adam
Loss Function	Categorical Cross-entropy	Categorical Cross-entropy	Categorical Cross-entropy	Categorical Cross-entropy	Categorical Cross-entropy

6. Result Analysis

This section presents the evaluation results of the performance of deep learning (DL) models mentioned above to the FL training scenario that incorporates the SSL mechanism on the ToN_IoT dataset. The key performance evaluation parameters include Accuracy, Precision, Recall, F1_score, Loss, and the application of the FedAvg aggregation algorithm. In our experimental setup, each model was applied to seven different IoT sensor datasets individually. The evaluation metrics for these selected models for the seven clients are presented in Table 4.

Table 4. Table of Evaluation Metrics Results.

Algorithms	Round No.	Clients	Accuracy	Precision	Recall	F1_Score	Loss
LeNet	50	C=1	0.9168	0.8775	0.8776	0.8775	0.2032
		C=2	0.9167	0.8774	0.8775	0.8774	0.2031
		C=3	0.9166	0.8775	0.8777	0.8775	0.2032
		C=4	0.9166	0.8777	0.8779	0.8778	0.2031
		C=5	0.9165	0.8774	0.8776	0.8775	0.2031
		C=6	0.9166	0.8775	0.8777	0.8776	0.2031
		C=7	0.9167	0.8776	0.8778	0.8777	0.2032
DNN	50	C=1	0.9130	0.8704	0.8706	0.8705	0.2158
		C=2	0.9128	0.8706	0.8708	0.8707	0.2157
		C=3	0.9127	0.8709	0.8710	0.8709	0.2157
		C=4	0.9131	0.8709	0.8711	0.8710	0.2152
		C=5	0.9126	0.8707	0.8708	0.8708	0.2153
		C=6	0.9129	0.8709	0.8711	0.8710	0.2151
		C=7	0.9127	0.8709	0.8711	0.8710	0.2157
LSTM	50	C=1	0.9061	0.8584	0.8586	0.8585	0.2380
		C=2	0.9061	0.8587	0.8589	0.8588	0.2376
		C=3	0.9062	0.8588	0.8589	0.8589	0.2386
		C=4	0.9061	0.8591	0.8592	0.8591	0.2379
		C=5	0.9061	0.8586	0.8589	0.8587	0.2379
		C=6	0.9061	0.8585	0.8586	0.8586	0.2381
		C=7	0.9062	0.8586	0.8589	0.8587	0.2380
GRU	50	C=1	0.9069	0.8619	0.8620	0.8620	0.2329
		C=2	0.9073	0.8618	0.8620	0.8619	0.2328
		C=3	0.9072	0.8618	0.8620	0.8619	0.2329
		C=4	0.9071	0.8617	0.8619	0.8618	0.2326
		C=5	0.9070	0.8616	0.8618	0.8617	0.2329
		C=6	0.9071	0.8613	0.8615	0.8614	0.2327
		C=7	0.9070	0.8615	0.8616	0.8616	0.2327
FCN	50	C=1	0.9154	0.8741	0.8747	0.8744	0.2093
		C=2	0.9150	0.8741	0.8747	0.8744	0.2093
		C=3	0.9147	0.8742	0.8744	0.8741	0.2093
		C=4	0.9151	0.8740	0.8746	0.8743	0.2092
		C=5	0.9148	0.8743	0.8749	0.8746	0.2093
		C=6	0.9147	0.8742	0.8747	0.8745	0.2093
		C=7	0.9153	0.8743	0.8749	0.8746	0.2092

6.1. Model Performance Analysis

Each DL model was applied to seven chosen IoT sensor datasets in such a way that each sensor was viewed individually as a client. These models were trained for 50 rounds on the data provided by each sensor. These sensors have been denoted as follows: C1 for the Fridge sensor, C2 for GPS Tracker sensor, C3 for Motion Light sensor, C4 for Garage Door sensor, C5 for Modbus sensor, C6 for Thermostat sensor, and C7 for Weather sensor. Table 4 provides a detailed overview of the optimal metric outcomes for each sensor dataset across various models.

The Lenet model demonstrated exceptional performance by achieving very high results, especially in the case of the Fridge sensor (C1), where the highest accuracy was achieved. LeNet's convolutional layers are to capture spatial features in order, so it is most suitable for detecting intrusions based on unusual spatial patterns. This capability is evident in its superior performance on the Fridge sensor (C1), where it outperformed other models. It also performed well in the Garage Door sensor (C4) with respect to precision, recall, and f1_score with the least loss recorded, which means that this model was able to capture patterns in this dataset with minimal number of errors.

The Fully Convolutional Network (FCN) model also performed well, particularly with the Weather sensor (C7), where it obtained the highest precision, recall, and f1_score at the lowest loss. This shows that FCN is able to perform at predicting accurate outcomes while maintaining class balance, particularly for the Weather sensor (C7). FCN is less likely to skew its predictions towards one class over another, ensuring that it doesn't falsely identify normal sensor readings as intrusions. The Weather sensor might have a unique distribution of intrusion and non-intrusion patterns. For example, sudden and extreme weather events might be more likely to trigger intrusion alarms. FCN's ability to balance predictions helps ensure that the model does not become overly sensitive to these events, leading to false alarms.

The Gated Recurrent Unit (GRU) model achieved excellent precision, recall, and f1_score for the Fridge sensor (C1) and highest accuracy for the GPS Tracker sensor (C2). The GPS Tracker sensor generates location data over time. Due to the capabilities of learning temporal dependencies in this data, GRU was able to detect unusual movement patterns that might indicate an intrusion. For the Garage Door sensor (C4), least loss was recorded, which means that GRU was quite accurate in its predictions with only minimal errors. It confirms that the GRU's gating mechanism enables it to learn long-term dependencies in the sequential data, rendering it suitable for the time-series data.

The Long Short-Term Memory (LSTM) model was able to minimize the loss percentage for GPS Tracker sensor (C2) and achieved the highest accuracy among all models for both the Motion Light (C3) and Weather (C7) sensors, suggesting its effectiveness in predicting the behavior of these sensors based on historical patterns. Motion patterns often have temporal dependencies, and weather patterns are highly dependent on past conditions and exhibit both short-term and long-term dependencies. LSTM's ability to remember information for long periods of time makes it well-suited for tasks that require understanding the context of temporal and past events.

On the other hand, the Deep Neural Network (DNN) model ranked best as it exhibited an overall superior performance for the Garage Door sensor (C4), obtaining the highest metrics across the board. DNNs are a versatile architecture that can be applied to a wide range of tasks. While they may not have the specialized strengths of other models, they often provide a robust baseline for intrusion detection tasks.

Within our analysis, Table 4 depicts that both Lenet and FCN displayed effectiveness with the Fridge sensor dataset, while LSTM excelled with Light and Weather sensor datasets, and the DNN model demonstrated proficiency with the Garage Door sensor dataset. Based on the metrics, GRU slightly outperformed LSTM in terms of accuracy and loss for the GPS Tracker sensor. However, none of the models achieved the pinnacle of accuracy for the Modbus (C5) and Thermostat (C6) sensor datasets. This comprehensive analysis underscores the sophisticated strengths of each model across diverse IoT and IIoT sensors. It means that the choice of the best ML or DL model for a particular IoT sensor application may depend on the specific sensor characteristics.

Table 5 highlights the top performance metrics for the seven clients in the chosen models. LeNet stands out with the highest overall performance, achieving an impressive 91.68% accuracy, 87.77% precision, 87.79% recall, 87.78% f1-score, and a loss of just 20.31% - the lowest recorded for any model. This indicates that the LeNet performs best overall in terms of both attcak detection capability and minimizing errors. FCN performs almost as well as LeNet, with a very close accuracy of 91.53% and relatively low loss (20.92%). Its precision, recall, and F1-score are also high, making it one of the top performers in terms of balancing accuracy and error minimization. DNN is slightly behind LeNet and

FCN in terms of accuracy (91.31%) with a slightly higher loss of 21.51%, indicating a marginally higher rate of error compared to LeNet. LSTM shows relatively lower performance compared to other models and has a higher loss of 23.76%, suggesting more errors in predictions. The GRU and LSTM algorithms exhibit similar performance, with GRU having a marginally higher accuracy of 90.73% compared to LSTM's 90.62%.

Table 5. FL algorithms' performance comparison metrics.

Algorithm	Accuracy	Precision	Recall	F1_Score	Loss
LeNet	0.9168	0.8777	0.8779	0.8778	0.2031
DNN	0.9131	0.8709	0.8711	0.8710	0.2151
LSTM	0.9062	0.8591	0.8592	0.8591	0.2376
GRU	0.9073	0.8619	0.8620	0.8620	0.2326
FCN	0.9153	0.8743	0.8749	0.8746	0.2092

6.2. Server Accuracy and Loss Trajectory

Figure 7 shows the server accuracy of all five models over the number of rounds, where the x-axis shows the number of rounds, and the y-axis shows the classification accuracy. The vertical line at round 40 marks the early stopping checkpoints for the models. Early stopping is a regularization technique used to prevent overfitting, where a model learns the training data too well and fails to generalize to unseen data.

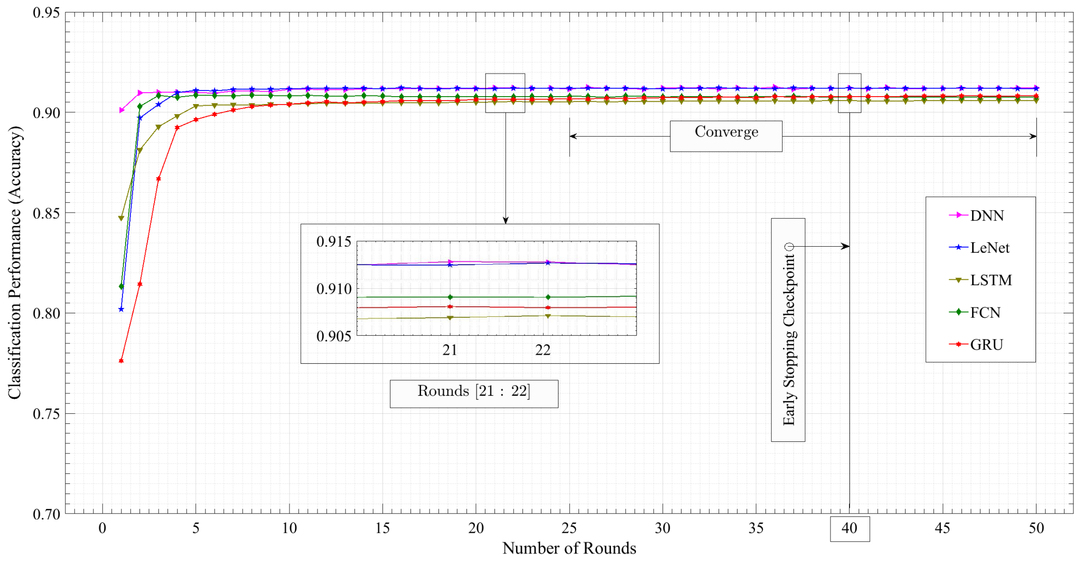

Figure 7. FL algorithms' server global validation accuracy.

The server accuracy for all five models increases over time, indicating that the models are learning. However, the rate of learning is different for each model. The DNN model demonstrates the fastest convergence, reaching the highest server accuracy at the end of 50 rounds, approximately 0.912188. The DNN's strong performance can be attributed to its ability to learn complex patterns and relationships within the data. However, it may be prone to overfitting if not carefully regularized. The LSTM model reaches a plateau in accuracy after around 20 rounds. While this model excels in capturing long-term dependencies, its performance stabilized early at a high level. This indicates that the model has effectively learned the patterns in the data and is not overfitting. LeNet and GRU models continue to improve up to round 30. The FCN model will likely overfit the training data by round 40, so its accuracy will likely decrease on new data. By halting training at round 40, this model is prevented from overfitting, ensuring better performance on new data.

Figure 8 provides a graphical representation of the server loss trajectory across multiple rounds for the five models. The x-axis denotes the progression of rounds, while the y-axis captures the classification loss incurred during each corresponding round. Notably, a vertical demarcation line at round 40 serves as a significant point, indicating the early stopping checkpoints for the respective models. This visual depiction offers a comprehensive overview of the models' loss dynamics and the strategic decision to halt training at a specific juncture, shedding light on the convergence patterns and optimization strategies employed.

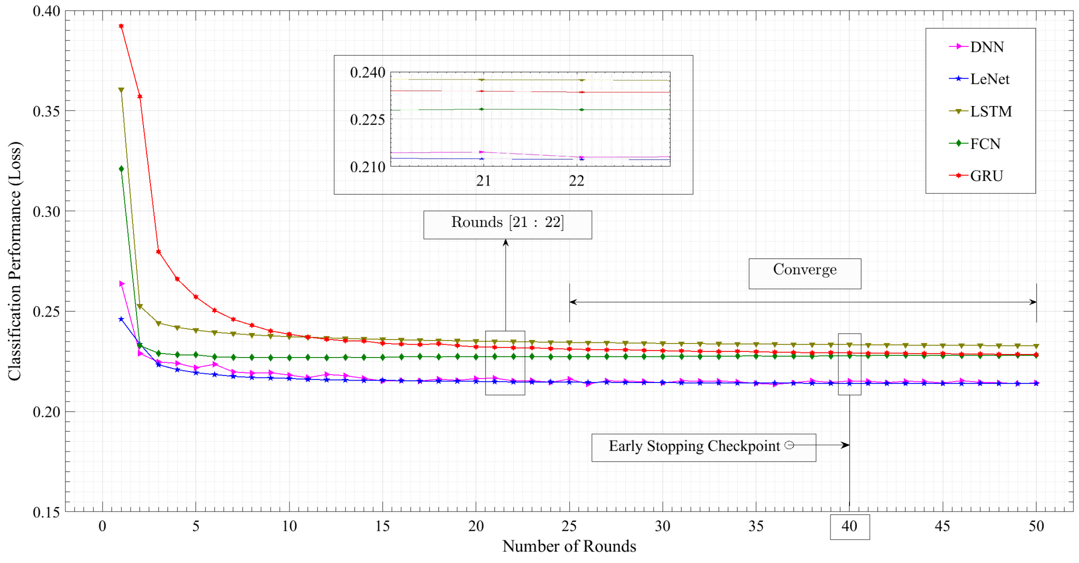

Figure 8. DFL algorithms' server global validation loss.

As time progresses, the server loss of all five models gradually decreases. The LeNet model exhibits the lowest server loss at the end of 50 rounds, approximately 0.213967. The FCN and LSTM models reach a point where their loss stabilizes after about 20 rounds, whereas the LeNet and GRU models continue to improve up to round 30. From round 5 to round 22, the DNN model shows notable variations in loss. However, following round 22, these fluctuations become less pronounced, and the loss value stabilizes.

7. Conclusion

This study represents a novel privacy-preserving federated learning based approach for detecting intrusion in heterogeneous IoT networks. By utilizing the collective insights of distributed IoT devices without putting data privacy at risk, our approach addresses the critical drawbacks of existing intrusion detection systems. The proposed architecture contains seven diverse IoT sensors as clients and a cloud server for global model aggregation, demonstrating the effectiveness of federated learning in increasing the security for IoT ecosystems. Our comprehensive evaluation compares five Deep Learning models across various types of IoT sensors, highlighting the potential of this approach. Among all these DL architectures we evaluated, the LeNet model achieved the highest accuracy of 91.68% along with balanced precision and recall. This indicates the capability of federated learning to effectively detect and classify a wide range of cyber threats. Moreover, different models showed variability in their performance in different sensor scenarios. It emphasizes the importance of tailored approaches for specific IoT applications. The integration of SSL for secure communication and the FedAvg algorithm for model aggregation further strengthens the robustness and reliability of our system. By maintaining data locality and enabling collaborative learning, this approach enhances scalability and efficiency in handling the growing complexity of IoT networks, along with preserving data privacy. Future work in this domain can build on these findings to further refine and expand federated learning applications in IoT security, contributing to safer and more reliable smart environments.

Author Contributions: Conceptualization, S.A.M, N.I., Z.I. and S.T.M.; Formal analysis, S.A.M. and N.I; Funding acquisition, Z.R. and S.T.M.; Investigation, S.A.M., N.I., and Z.I.; Methodology, S.A.M., N.I., and S.T.M.; Supervision, S.T.M. and Z.R.; Validation, S.T.M, Z.R.; Visualization, N.I, S.A.M., and S.T.M.; Writing—original draft, S.A.M, N.I. and Z.I.; Writing—review and editing, S.T.M, and Z.R. All authors have read and agreed to the published version of the manuscript.

Informed Consent Statement: All information used in this study was sourced from publicly available articles and databases, and therefore, informed consent and ethical approval were not required.

Data Availability Statement: The data presented in this manuscript are available on request.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ANN	Artificial Neural Network
CDL	Centralized Deep Learning
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CPS	Cyber-Physical Systems
DDL	Distributed Deep Learning
DDoS	Distributed Denial of Service
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
FCL	Fully Connected Layer
FCN	Fully Convolutional Network
FedAGRU	Federated Learning based Attention Gated Recurrent Uni
FedAvg	Federated Averaging
FDL	Federated Deep Learning
FL	Federated Learning
FSL	Federated Sequence Learning
GRU	Gated Recurrent Unit
GPU	Graphics Processing Unit
ICS	Industrial Control Systems
IDS	Intrusion Detection System
IIoT	Industrial Internet of Things
IoT	Internet of Things
JC	Jungle Computing
KMA	Key Management Authority
LDL	localized Deep Learning
LeNet	LeCun Network
LSTM	Long Short-Term Memory
ML	Machine Learning
MITM	Man-in-the-Middle
NIDS	Network Intrusion Detection System
PCA	Password Cracking Attack
PHEC	Probabilistic Hybrid Ensemble Classification
ReLU	Rectified Linear Unit
SSL	Secure Sockets Layer
STIN	Satellite Terrestrial Integrated Networks
SVDD	Support Vector Data Description
VAE	Variational Autoencoder
XAI	Explainable Artificial Intelligence
XSS	Cross-Site Scripting

References

1. Mehedi, S.; Anwar, A.; Rahman, Z.; Ahmed, K.; Islam, R. Dependable Intrusion Detection System for IoT: A Deep Transfer Learning-Based Approach. *IEEE Trans. Ind. Inf*, *19*, 1006–1017,. doi:10.1109/TII.2022.3164770.
2. Alsaedi, A.; Moustafa, N.; Tari, Z.; Mahmood, A.; Anwar, A. TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. *IEEE Access*, *8*, 165130–165150,. doi:10.1109/ACCESS.2020.3022862.
3. Konečný, J.; McMahan, H.; Yu, F.; Richtárik, P.; Suresh, A.; Bacon, D.L. Federated Learning: Strategies for Improving Communication Efficiency. *Strategies for Improving Communication Efficiency*.
4. Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 3rd ed.; United States of America: O'Reilly Media, Inc, 2022.
5. Lim, W.; Luong, N.; Hoang, D.; Jiao, Y.; Liang, Y.C.; Yang, Q.; Niyato, D.; Miao, C. Federated Learning in Mobile Edge Networks: A Comprehensive Survey. *IEEE Commun. Surv. Tutorials*, *22*, 2031–2063,. doi:10.1109/COMST.2020.2986024.
6. Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; Chandra, V. Federated Learning with Non-IID Data. doi:10.48550/arXiv.1806.00582.
7. Bhuiyan, M.; Kuo, S.; Lyons, D.; Shao, Z. Dependability in Cyber-Physical Systems and Applications. *ACM Trans. Cyber-Phys. Syst*, *3*, 1–4,. doi:10.1145/3271432.
8. Aledhari, M.; Razzak, R.; Parizi, R.; Saeed, F. Federated Learning: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Access*, *8*, 140699–140725,. doi:10.1109/ACCESS.2020.3013541.
9. Li, T.; Sahu, A.; Talwalkar, A.; Smith, V. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag*, *37*, 50–60,. doi:10.1109/MSP.2020.2975749.
10. Xu, J.; Glicksberg, B.; Su, C.; Walker, P.; Bian, J.; Wang, F. Federated Learning for Healthcare Informatics. *J Healthc Inform Res*, *5*, 1–19,. doi:10.1007/s41666-020-00082-4.
11. Wu, Q.; He, K.; Chen, X. Personalized Federated Learning for Intelligent IoT Applications: A Cloud-Edge Based Framework. *IEEE Open J. Comput. Soc*, *1*, 35–44,. doi:10.1109/OJCS.2020.2993259.
12. Li, B.; Wu, Y.; Song, J.; Lu, R.; Li, T.; Zhao, L. Federated Deep Learning for Intrusion Detection in Industrial Cyber–Physical Systems. *IEEE Trans. Ind. Inf*, *17*, 5615–5624,. doi:10.1109/TII.2020.3023430.
13. Yuan, B.; Ge, S.; Xing, W. A Federated Learning Framework for Healthcare IoT Devices.
14. Billah, M.; Mehedi, S.; Anwar, A.; Rahman, Z.; Islam, R. A Systematic Literature Review on Blockchain Enabled Federated Learning Framework for Internet of Vehicles.
15. Lo, S.; Lu, Q.; Wang, C.; Paik, H.Y.; Zhu, L. A Systematic Literature Review on Federated Machine Learning: From a Software Engineering Perspective. *ACM Comput. Surv*, *54*, 1–39,. doi:10.1145/3450288.
16. Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečný, J.; Mazzocchi, S.; McMahan, H. Towards Federated Learning at Scale: System Design 2019.
17. Li, Z.; Sharma, V.; Mohanty, P.; S.. Preserving Data Privacy via Federated Learning: Challenges and Solutions. *IEEE Consumer Electron. Mag*, *9*, 8–16,. doi:10.1109/MCE.2019.2959108.
18. Neranjan Thilakarathne, N.; Muneeswari, G.; Parthasarathy, V.; Alassery, F.; Hamam, H.; Kumar Mahendran, R.; Shafiq, M. Federated Learning for Privacy-Preserved Medical Internet of Things. *Intelligent Automation & Soft Computing*, *33*, 157–172,. doi:10.32604/iasc.2022.023763.
19. Khan, L.; Saad, W.; Han, Z.; Hossain, E.; Hong, C. Federated Learning for Internet of Things: Recent Advances. *Taxonomy, and Open Challenges*.
20. Fang, H.; Qian, Q. Privacy Preserving Machine Learning with Homomorphic Encryption and Federated Learning. *Future Internet*, *13*, 94,. doi:10.3390/fi13040094.
21. Ruzafa-Alcazar, P.; Fernandez-Saura, P.; Marmol-Campos, E.; Gonzalez-Vidal, A.; Hernandez-Ramos, J.; Bernal-Bernabe, J.; Skarmeta, A. Intrusion Detection Based on Privacy-Preserving Federated Learning for the Industrial IoT. *IEEE Trans. Ind. Inf*, *19*, 1145–1154,. doi:10.1109/TII.2021.3126728.
22. Chatterjee, S.; Hanawal, M. Federated Learning for Intrusion Detection in IoT Security: A Hybrid Ensemble Approach. *IJITCA*, *2*, 62,. doi:10.1504/IJITCA.2022.124372.
23. Lazzarini, R.; Tianfield, H.; Charissis, V. Federated Learning for IoT Intrusion Detection. *AI*, *4*, 509–530,. doi:10.3390/ai4030028.

24. Zhang, T.; He, C.; Ma, T.; Gao, L.; Ma, M.; Avestimehr, S. Federated Learning for Internet of Things. Proceedings of the Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems; Coimbra Portugal: ACM, 2021; p. 413–419.
25. Chen, Z.; Lv, N.; Liu, P.; Fang, Y.; Chen, K.; Pan, W. Intrusion Detection for Wireless Edge Networks Based on Federated Learning. *IEEE Access*, *8*, 217463–217472,. doi:10.1109/ACCESS.2020.3041793.
26. Aashmi, S.; R., J.; T.. Intrusion Detection Using Federated Learning for Computing. *Computer Systems Science and Engineering*, *45*, 1295–1308,. doi:10.32604/csse.2023.027216.
27. Huong, T.; Bac, T.; Ha, K.; Hoang, N.; Hoang, N.; Hung, N.; Tran, K. Federated Learning-Based Explainable Anomaly Detection for Industrial Control Systems. *IEEE Access*, *10*, 53854–53872,. doi:10.1109/ACCESS.2022.3173288.
28. Popoola, S.; Ande, R.; Adebisi, B.; Gui, G.; Hammoudeh, M.; Jogunola, O. Federated Deep Learning for Zero-Day Botnet Attack Detection in IoT-Edge Devices. *IEEE Internet Things J*, *9*, 3930–3944,. doi:10.1109/JIOT.2021.3100755.
29. Li, K.; Zhou, H.; Tu, Z.; Wang, W.; Zhang, H. Distributed Network Intrusion Detection System in Satellite-Terrestrial Integrated Networks Using Federated Learning. *IEEE Access*, *8*, 214852–214865,. doi:10.1109/ACCESS.2020.3041641.
30. Li, F.; Lin, J.; Han, H. FSL: Federated Sequential Learning-Based Cyberattack Detection for Industrial Internet of Things. *Industrial Artificial Intelligence*, *1*, 4,. doi:10.1007/s44244-023-00006-2.
31. Sarhan, M.; Layeghy, S.; Moustafa, N.; Portmann, M. Cyber Threat Intelligence Sharing Scheme Based on Federated Learning for Network Intrusion Detection. *J Netw Syst Manage*, *31*, 3,. doi:10.1007/s10922-022-09691-3.
32. Doriguzzi-Corin, R.; Siracusa, D. FLAD: Adaptive Federated Learning for DDoS Attack Detection. *Computers & Security*, *137, 103597*. doi:10.1016/j.cose.2023.103597.
33. Jurcut, A.; Niculcea, T.; Ranaweera, P.; Le-Khac, N.A. Security Considerations for Internet of Things: A Survey. SN COMPUT. *SCI*, *1, 193*. doi:10.1007/s42979-020-00201-3.
34. Sisinni, E.; Saifullah, A.; Han, S.; Jennehag, U.; Gidlund, M. Industrial Internet of Things: Challenges, Opportunities, and Directions. *IEEE Trans. Ind. Inf*, *14*, 4724–4734,. doi:10.1109/TII.2018.2852491.
35. Xu, L.; He, W.; Li, S. Internet of Things in Industries: A Survey. *IEEE Trans. Ind. Inf*, *10*, 2233–2243,. doi:10.1109/TII.2014.2300753.
36. Moshawrab, M.; Adda, M.; Bouzouane, A.; Ibrahim, H.; Raad, A.F.L.A.A.; Strategies, C. Reviewing Federated Learning Aggregation Algorithms; Strategies, Contributions, Limitations and Future Perspectives. *Limitations and Future Perspectives. Electronics*, *12*, 2287,. doi:10.3390/electronics12102287.
37. Zhang, C.; Chen, Y.; Meng, Y.; Ruan, F.; Chen, R.; Li, Y.; Yang, Y. A Novel Framework Design of Network Intrusion Detection Based on Machine Learning Techniques. *Security and Communication Networks*, pp. 1–15,. doi:10.1155/2021/6610675.
38. Dastres, R.; Soori, M. Secure Socket Layer (SSL) in the Network and Web Security. *International Journal of Computer and Information Engineering*, *14*, 330–333. In press,.
39. Singh, A.; Loar, R. Web Security and Enhancement Using SSL : A Review. *International Journal of Scientific Research in Science and Technology*, p. 4.
40. Das, M.; Samdaria, N. On the Security of SSL/TLS-Enabled Applications. *Applied Computing and Informatics*, *10*, 68–81,. doi:10.1016/j.aci.2014.02.001.
41. Roy, P.; Singh, J.; Banerjee, S. Deep Learning to Filter SMS Spam. *Future Generation Computer Systems*, *102*, 524–533,. doi:10.1016/j.future.2019.09.001.
42. Nisioti, A.; Mylonas, A.; Yoo, P.; Katos, V. From Intrusion Detection to Attacker Attribution: A Comprehensive Survey of Unsupervised Methods. *IEEE Commun. Surv. Tutorials*, *20*, 3369–3388,. doi:10.1109/COMST.2018.2854724.
43. Tampuu, A.; Bzhalava, Z.; Dillner, J.; Vicente, R. ViraMiner: Deep Learning on Raw DNA Sequences for Identifying Viral Genomes in Human Samples.
44. Li, X.; Hu, Z.; Xu, M.; Wang, Y.; Ma, J. Transfer Learning Based Intrusion Detection Scheme for Internet of Vehicles. *Information Sciences*, *547*, 119–135,. doi:10.1016/j.ins.2020.05.130.
45. Guyon, I. A Scaling Law for the Validation-Set Training-Set Size Ratio. *AT & T Bell Laboratories*, p. 1.